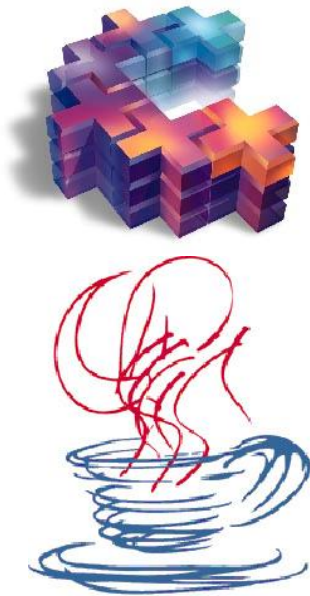
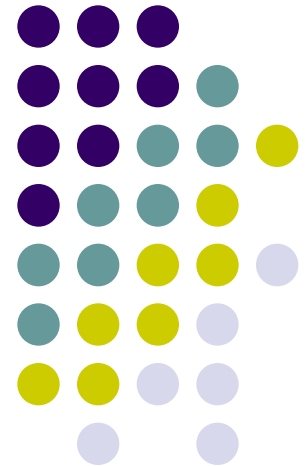


TECHNIQUES DE COMPILATION

CH2: ANALYSE SYNTAXIQUE



Olfa MOUELHI
olfa.mouelhi@esprit.tn
esprit 
Ecole Supérieure Privée
d'Ingénierie et de Technologies

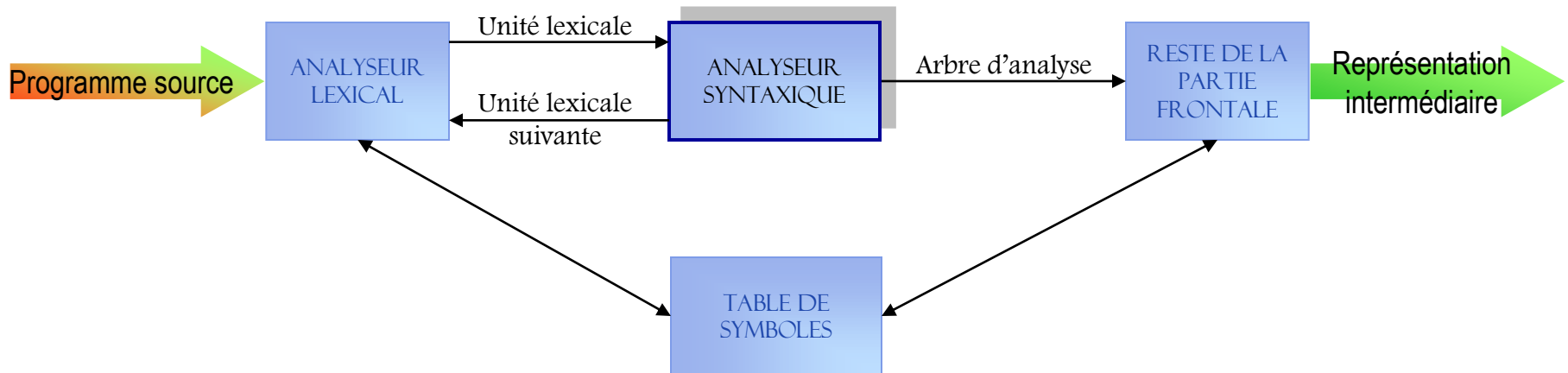


INTRODUCTION

- But de l'analyse syntaxique : transformer un flot d'unités lexicales en arbre abstrait.
- Langage de programmation : règles prescrivant la structure syntaxique des programmes bien formés.
- Langage C , Programme = ensemble de Fonctions.
 - Fonction : formée de blocs.
 - Un bloc : formé d'instructions.
 - Une instruction : formée d'expressions.
 - Une expression : formée d'unités lexicales.
 - Etc.
- La syntaxe d'un langage peut être décrite par des grammaires non contextuelles (notation BNF – Backus-Naur Form).
- Grammaire : spécification syntaxique précise et facile à comprendre d'un langage de programmation.

RÔLE DE L'ANALYSEUR SYNTAXIQUE

- L'analyseur syntaxique obtient une chaîne d'unités lexicales de l'analyseur lexical.
- Il vérifie que la chaîne peut être engendrée par la grammaire du langage source.
- Il signale chaque erreur de syntaxe de manière intelligible.
- Supporte les erreurs les plus communes de façon à pouvoir continuer l'analyse du code restant.



RÔLE DE L'ANALYSEUR SYNTAXIQUE

- Deux types d'analyses sont retenues pour les compilateurs : Ascendante ou Descendante.
- Analyseurs descendants construisent les arbres d'analyse de haut (ROOT) en bas (LEAF).
- Analyseurs ascendants partent des feuilles et remontent à la racine.
- Dans les deux cas l'entrée de l'analyse syntaxique est explorée par convention de la gauche vers la droite un symbole à la fois.
- Le gestionnaire d'erreurs doit :
 - Indiquer la présence d'erreurs de façon claire et précise.
 - Traiter chaque erreur pour pouvoir détecter les erreurs suivantes.
- Le gestionnaire d'erreurs ne doit pas pénaliser l'exécution des programmes correctes.

GRAMMAIRE NON CONTEXTUELLE

- Permet d'exprimer des règles du type
Si S_1 et S_2 sont des Instructions et E une Expression alors :
« **if E then S_1 else S_2** » est une instruction
- Cette forme d'instructions ne peut être spécifiée par des ER.
- Si «*instruction*» est la variable syntaxique dénotant la classe des instructions et «*expression*» la variable syntaxique dénotant la classe des expressions; alors cette forme d'instructions peut être exprimée de manière claire et lisible par la règle de production :

instruction \rightarrow **if** *expression* **then** *instruction* **else** *instruction*

- Une grammaire non contextuelle est composée de :
 - Terminaux : les symboles de base pouvant former les chaînes (\cong unités lexicales).
 - Non terminaux : variables syntaxiques dénotant un ensemble de chaînes (*expression*, *instruction*)
 - Axiome : Un non terminal particulier qui dénote l'ensemble des chaînes du langage (départ).
 - Les productions de la grammaire : comment combiner les terminaux et les non terminaux (règles).

GRAMMAIRE NON CONTEXTUELLE

La grammaire constituée des productions suivantes définit des expressions arithmétiques simples :

```
expression → expression OP expression
expression → ( expression )
expression → - expression
expression → id
      OP → +
      OP → -
      OP → *
      OP → /
      OP → ↑
```

Symboles terminaux : **id** + - * / ↑ ()

Symboles non terminaux : expression et OP

DÉRIVATION

- La dérivation donne une idée précise de la construction descendante de l'arbre d'analyse.
- Traiter une production comme une règle de réécriture : le non terminal de la partie gauche est remplacé par la chaîne en partie de droite de la production.

Considérons la grammaire suivante (E est une expression) :

$$E \rightarrow E + E \mid E * E \mid (E) \mid - E \mid \mathbf{id} \quad (G1)$$

La production : $E \rightarrow - E$ indique qu'une expression précédée du signe moins ($-$) est aussi une expression.

Cette expression peut engendrer des expressions plus complexes à partir d'expressions plus simples (remplacer une instance de E par $- E$).

$$E \Rightarrow - E \text{ (Se lit } E \text{ se dérive en } - E \text{)}$$

DÉRIVATION

- $E \rightarrow (E)$: possibilité de remplacer une instance d'un E dans une chaîne de symboles grammaticaux par (E) ; par exemple $E * E \Rightarrow (E) * E$ ou encore $E * E \Rightarrow E * (E)$.
- On peut prendre un E unique et appliquer répétitivement les productions dans un ordre quelconque :
- $E \Rightarrow - E \Rightarrow - (E) \Rightarrow - (\mathbf{id})$
- On dit que $\alpha A \beta \Rightarrow \alpha \gamma \beta$ si :
 - $A \rightarrow \gamma$
 - α et β sont des chaînes arbitraires de symboles grammaticaux.
- $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \alpha_n$ on dit que α_1 se dérive en α_n
- Le symbole \Rightarrow veut dire dérivation en une étape.
- Le symbole \Rightarrow^* veut dire dérivation en zéro, une ou plusieurs étapes.
- $\alpha \Rightarrow^* \alpha, \forall \alpha$ une chaîne
- Si $\alpha \Rightarrow^* \beta$ et $\beta \Rightarrow \gamma$ alors $\alpha \Rightarrow^* \gamma$.
- Le symbole \Rightarrow^+ veut dire dérivation en une ou plusieurs étapes.

DÉRIVATION

Soit G une grammaire et S son axiome. $\mathcal{L}(G)$ est le langage engendré par G . Les chaînes de $\mathcal{L}(G)$ contiennent uniquement les symboles terminaux de G . w une chaîne de terminaux.

- $w \in \mathcal{L}(G) \Leftrightarrow S \xRightarrow{+} w$.
- w est phrase de G .

- Un langage qui peut être engendré par une grammaire est dit langage non contextuel.
- Deux grammaires qui engendrent le même langage sont dites équivalentes.
- Si $S \xRightarrow{*} \alpha$, où α peut contenir certains non terminaux, α est une proto-phrase de G .
- Une phrase de G est une proto-phrase de G ne contenant aucun non-terminal.

EXEMPLE : la chaîne $-(\mathbf{id} + \mathbf{id})$ est une phrase de la grammaire G_1 (page 7) car on a la dérivation :

$$E \Rightarrow - E \Rightarrow - (E) \Rightarrow - (E + E) \Rightarrow - (\mathbf{id} + E) \Rightarrow - (\mathbf{id} + \mathbf{id})$$

$$E \xRightarrow{*} - (\mathbf{id} + \mathbf{id})$$

DÉRIVATION

$$(1) E \Rightarrow - E \Rightarrow - (E) \Rightarrow - (E + E) \Rightarrow - (\mathbf{id} + E) \Rightarrow - (\mathbf{id} + \mathbf{id})$$

$$(2) E \Rightarrow - E \Rightarrow - (E) \Rightarrow - (E + E) \Rightarrow - (E + \mathbf{id}) \Rightarrow - (\mathbf{id} + \mathbf{id})$$

Bien que les deux dérivations aboutissent au même résultat, l'ordre n'est pas le même.

Considérons les dérivations dans lesquelles seuls le non-terminal le plus à gauche est remplacé à chaque étape (dérivations gauches).

$\alpha \xRightarrow{g} \beta$ indique une étape de dérivation dans laquelle le non terminal le plus à gauche de α a été remplacé.

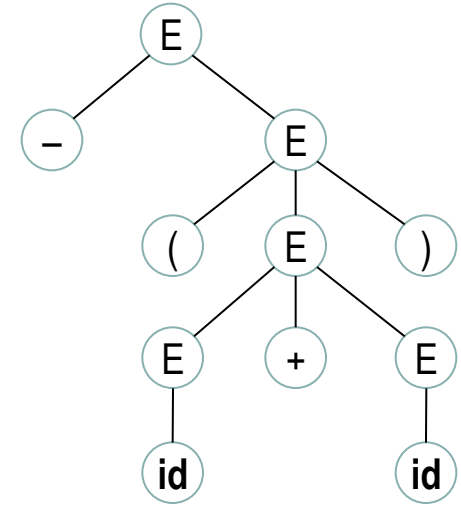
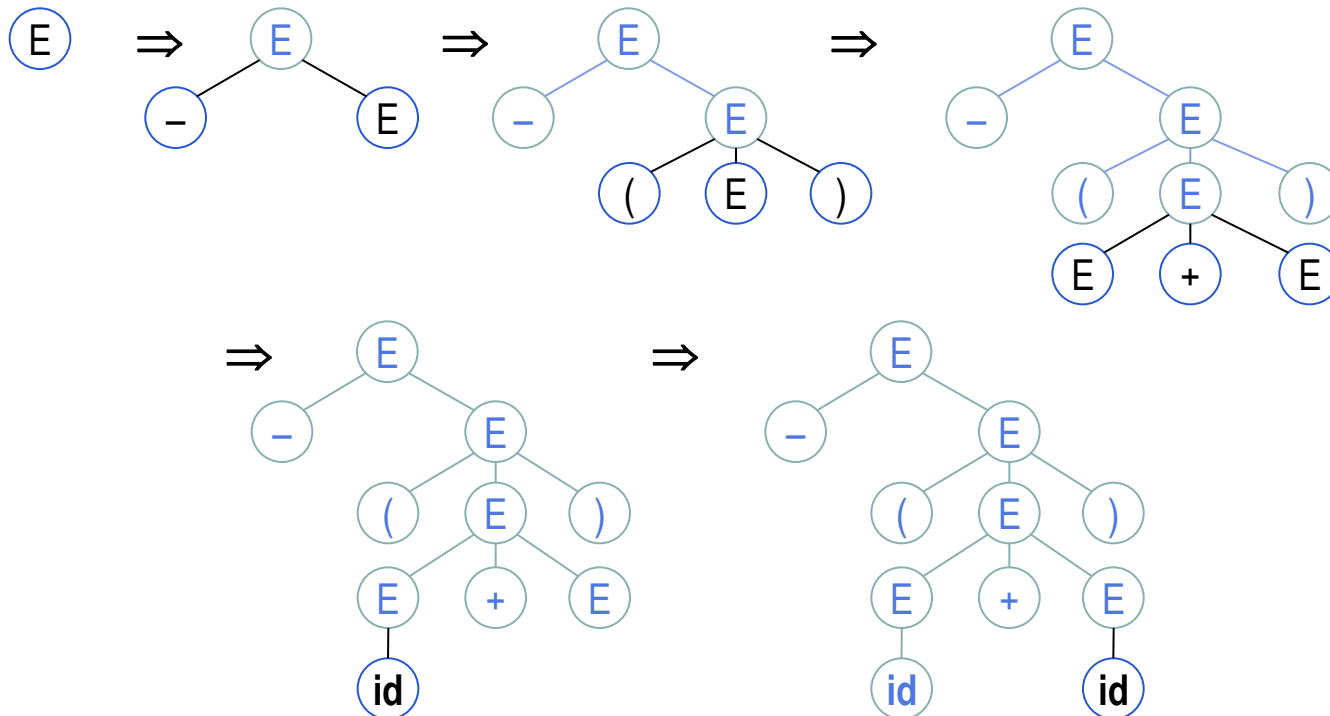
$$E \xRightarrow{g} - E \xRightarrow{g} - (E) \xRightarrow{g} - (E + E) \xRightarrow{g} - (\mathbf{id} + E) \xRightarrow{g} - (\mathbf{id} + \mathbf{id})$$

- $\alpha \xRightarrow{*g} \beta$: α se dérive en β par une dérivation gauche.
- $S \xRightarrow{*g} \alpha$, on dit que α est une proto-phrase gauche de la grammaire considérée.

REPRÉSENTATION GRAPHIQUE DE DÉRIVATION

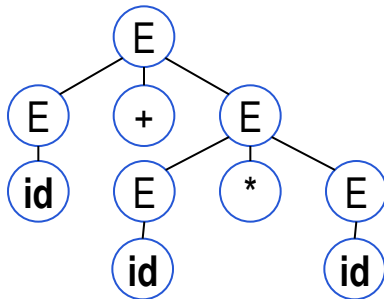
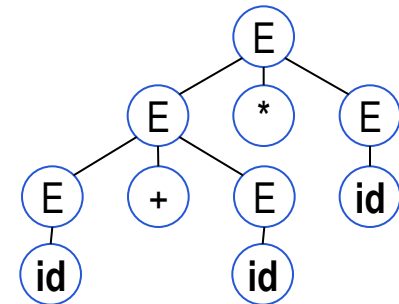
Un arbre d'analyse peut représenter une dérivation (le choix concernant l'ordre de remplacement ayant disparu).

RELATION ENTRE DÉRIVATION ET ARBRE D'ANALYSE:



REPRÉSENTATION GRAPHIQUE DE DÉRIVATION

Considérons la dérivation de la phrase **id + id * id** :

$$\begin{aligned} E &\Rightarrow E + E \\ &\Rightarrow \mathbf{id} + E \\ &\Rightarrow \mathbf{id} + E * E \\ &\Rightarrow \mathbf{id} + \mathbf{id} * E \\ &\Rightarrow \mathbf{id} + \mathbf{id} * \mathbf{id} \end{aligned}$$

$$\begin{aligned} E &\Rightarrow E * E \\ &\Rightarrow E + E * E \\ &\Rightarrow \mathbf{id} + E * E \\ &\Rightarrow \mathbf{id} + \mathbf{id} * E \\ &\Rightarrow \mathbf{id} + \mathbf{id} * \mathbf{id} \end{aligned}$$


La phrase **id + id * id** a deux dérivations gauches distinctes.

AMBIGUÏTÉ

- Une grammaire qui produit plus d'un arbre d'analyse pour une phrase donnée est dite ambiguë.
- Une grammaire ambiguë produit plus d'une dérivation à gauche ou plus d'une dérivation à droite pour la même phrase.
- Il est généralement nécessaire de rendre la grammaire non ambiguë.

Exemple:

$$E \rightarrow E+E \mid E^*E \mid (E) \mid x_1 \mid x_2$$

Cette grammaire est ambiguë car la chaîne w a plus d'une dérivation gauche:

$$E \rightarrow \underline{E}+E \rightarrow x_1+\underline{E} \rightarrow x_1+\underline{E}^*E \rightarrow x_1+x_1^*\underline{E} \rightarrow x_1+x_1^*x_2$$

$$E \rightarrow \underline{E}^*E \rightarrow \underline{E}+E^*E \rightarrow x_1+\underline{E}^*E \rightarrow x_1+x_1^*\underline{E} \rightarrow x_1+x_1^*x_2$$

De même on peut trouver deux dérivations droites

ÉCRITURE D'UNE GRAMMAIRE

- Les grammaires peuvent spécifier la syntaxe de la plupart des langages de programmation.
- Chaque construction décrite par une expression régulière peut être décrite par une grammaire.

EXEMPLE : l'expression régulière $(a \mid b)^*abb$ et la grammaire suivante :

$$A_0 \rightarrow aA_0 \mid bA_0 \mid aA_1$$

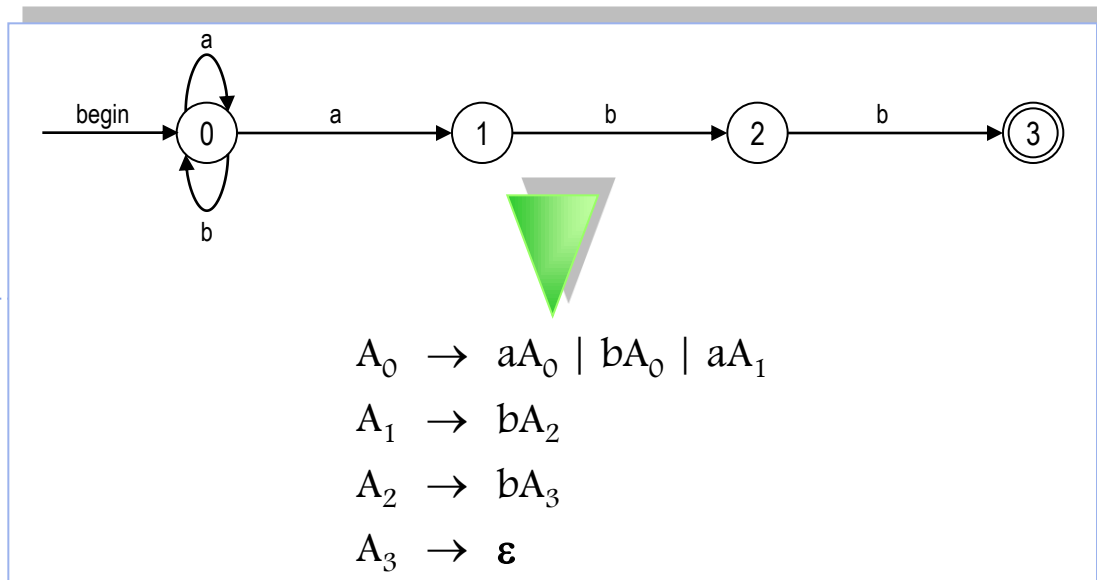
$$A_1 \rightarrow bA_2$$

$$A_2 \rightarrow bA_3$$

$$A_3 \rightarrow \epsilon$$

TRANSFORMATION D'UN AFN EN GRAMMAIRE

- Pour chaque état i de l'AFN on attribue un symbole non terminal A_i .
- Si l'état i a une transition vers l'état j sur le symbole a , on introduit la production :
 $A_i \rightarrow aA_j$
- Si l'état i a une transition vers l'état j sur le symbole ϵ , on introduit la production :
 $A_i \rightarrow A_j$
- Si i est l'état final, on introduit la production :
 $A_i \rightarrow \epsilon$



RÉCURSIVITÉ À GAUCHE ET À DROITE

- Une grammaire est récursive si elle contient un non terminal A qui se dérive en une production contenant A .
- Une production d'une grammaire est récursive à gauche si le symbole le plus à gauche de la partie droite de la production est le même que le non terminal de la partie gauche de la production (il existe une dérivation $A \xRightarrow{+} A\alpha$ (α une chaîne quelconque)).

EXEMPLE : $A \rightarrow A\alpha$

- Une production d'une grammaire est récursive à droite si le symbole le plus à droite de la partie droite de la production est le même que le non terminal de la partie gauche de la production.

EXEMPLE : $A \rightarrow \alpha A$

α est une suite de terminaux et non terminaux ne contenant pas A .

SUPPRESSION DE LA RÉCURSIVITÉ À GAUCHE

- Les méthodes d'analyse descendante ne fonctionnent pas avec les grammaires récursives à gauche.
- L'analyseur par descente récursive peut boucler indéfiniment.
- Pour les analyseurs descendants, il faut éliminer la récursivité à gauche.
- Des règles de réécriture pour éliminer la récursivité à gauche.
- A est non terminal dans les deux productions suivantes :

$$A \rightarrow A\alpha \mid \beta$$

α et β sont des suites de terminaux et non terminaux ne contenant pas A .

EXEMPLE : $expression \rightarrow expression + term \mid term$

SUPPRESSION DE LA RÉCURSIVITÉ À GAUCHE IMMÉDIATE

$$A \rightarrow A\alpha \mid \beta$$

Application répétée de cette production :

$$A \Rightarrow A\alpha \Rightarrow (A\alpha)\alpha \Rightarrow ((A\alpha)\alpha)\alpha \Rightarrow \dots \Rightarrow \beta\alpha\alpha\alpha \dots \alpha$$

On peut obtenir le même effet en réécrivant la production définissant A de la manière suivante :

$$A \rightarrow \beta R$$

$$R \rightarrow \alpha R \mid \varepsilon$$

R est le nouveau non terminal

La production $R \rightarrow \alpha R$ est récursive à droite.

SUPPRESSION DE LA RÉCURSIVITÉ À GAUCHE IMMÉDIATE

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid \text{id}$$

Élimination de récursivités à gauche immédiates (productions de la forme $A \rightarrow A\alpha$) :

$$E \rightarrow T E'$$

$$E' \rightarrow + T E' \mid \varepsilon$$

$$T \rightarrow F T'$$

$$T' \rightarrow * F T' \mid \varepsilon$$

$$F \rightarrow (E) \mid \text{id}$$

SUPPRESSION DE LA RÉCURSIVITÉ À GAUCHE CACHÉE

RÈGLES POUR L'ÉLIMINATION DE LA RÉCURSIVITÉ À GAUCHE IMMÉDIATE

- Grouper les A -productions comme suit :

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \dots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \dots \mid \beta_n \quad \text{Aucun } \beta_i \text{ ne commence par un } A$$

- On remplace les A -productions par :

$$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \dots \mid \beta_n A'$$

$$A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \dots \mid \alpha_m A' \mid \varepsilon$$

- Cet algorithme élimine toutes les récursivités à gauche immédiates.
- Il n'élimine pas les récursivités à gauche impliquant des dérivations contenant au moins deux étapes.

$$S \rightarrow Aa \mid B$$

$$A \rightarrow Ac \mid Sd \mid \varepsilon$$

Le non terminal S est récursif à gauche (mais pas immédiatement) car

$$S \Rightarrow Aa \Rightarrow Sda$$

SUPPRESSION DE LA RÉCURSIVITÉ À GAUCHE CACHÉE

ALGORITHME D'ÉLIMINATION DE LA RÉCURSIVITÉ À GAUCHE

DONNÉES : Une grammaire G sans cycles et sans productions vides

RÉSULTAT : Une grammaire équivalente à G sans récursivité à gauche

BEGIN

Ordonner les non terminaux A_1, A_2, \dots, A_n ;

FOR $i \leftarrow 1$ **TO** n **DO**

FOR $j \leftarrow 1$ **TO** $i - 1$ **DO**

 remplacer chaque production de la forme $A_i \rightarrow A_j \gamma$ par les productions

$A_i \rightarrow \delta_1 \gamma \mid \delta_2 \gamma \mid \dots \mid \delta_k \gamma$ ($A_j \rightarrow \delta_1 \mid \delta_2 \mid \dots \mid \delta_k$ sont des productions courantes)

END FOR

 éliminer les récursivités immédiates des A_i -productions

END FOR

END

SUPPRESSION DE LA RÉCURSIVITÉ À GAUCHE CACHÉE

EXEMPLE :

$$S \rightarrow Aa \mid b$$

$$A \rightarrow Ac \mid Sd \mid \varepsilon$$

- On classe les non terminaux dans l'ordre S, A
- Il n'y a pas de récursivité à gauche immédiate dans les S -productions. Rien ne se produit durant l'exécution de l'étape 2 pour $i = 1$.
- Pour $i = 2$, on substitue les S -productions dans $A \rightarrow Sd$ pour obtenir les A -productions suivantes :

$$A \rightarrow Ac \mid Aad \mid bd \mid \varepsilon$$

- En éliminant les récursivités à gauche immédiates, on obtient la grammaire suivante :

$$S \rightarrow Aa \mid b$$

$$A \rightarrow bdA' \mid A'$$

$$A' \rightarrow cA' \mid adA' \mid \varepsilon$$

FACTORISATION À GAUCHE

- Si plusieurs alternatives dans une production laquelle choisir ?
- Différer le choix jusqu'à ce que assez de texte source soit lu pour faire le bon choix.

```
instruction → if expression then instruction else instruction  
            | if expression then instruction
```

$A \rightarrow \alpha\beta_1 \mid \alpha\beta_2$ (Production de départ)

$A \rightarrow \alpha A'$

$A' \rightarrow \beta_1 \mid \beta_2$

FACTORISATION À GAUCHE

ALGORITHME FACTORISATION À GAUCHE

DONNÉES : Une grammaire G sans cycles et sans productions vides

RÉSULTAT : Une grammaire équivalente à G factorisée à gauche

Pour chaque non terminal A trouver le plus long préfixe α au moins à deux de ses alternatives.

Si $\alpha \neq \varepsilon$ alors remplacer toutes les A -productions :

$$A \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \dots \mid \alpha\beta_n \mid \gamma$$

γ Représente toutes les alternatives ne commençant pas par α par :


$$A \rightarrow \alpha A' \mid \gamma$$

$$A' \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$$

Répéter jusqu'à ce qu'aucune des alternatives d'un même non-terminal n'ait de préfixe commun.

FACTORISATION À GAUCHE

EXEMPLE :

$S \rightarrow iEtS \mid iEtSeS \mid a$		$S \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \gamma$ avec $\alpha = iEtS$, $\beta_1 = \varepsilon$, $\beta_2 = eS$ et $\gamma = a$
$E \rightarrow b$		
$S \rightarrow iEtSS' \mid a$		$S \rightarrow \alpha A' \mid \gamma$
$S' \rightarrow eS \mid \varepsilon$		$A' \rightarrow eS \mid \varepsilon$
$E \rightarrow b$		

À la vue de i , S peut être développé en $iEtSS'$ et attendre jusqu'à ce que $iEtS$ ait été reconnu pour développer S' en eS ou en ε

ANALYSE SYNTAXIQUE DESCENDANTE

ANALYSEURS PRÉDICTIFS

- Étant donné un symbole a en entrée et un non terminal A à développer, déterminer si une alternative de la production $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$ est l'unique alternative pouvant dériver en une chaîne commençant par a .

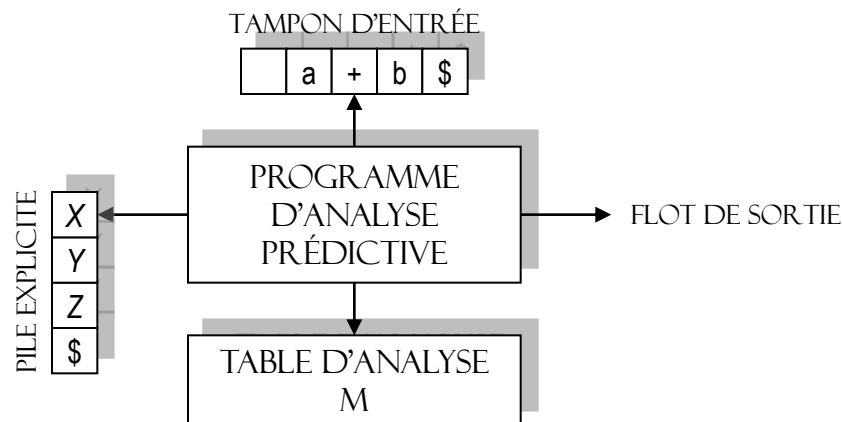
```
instruction → if expression then instruction else instruction  
            | while expression do instruction  
            | begin instructions_list end
```

Dans ce cas de figure, les mots clés **if**, **while** et **begin** nous prédisent quelle alternative est à prendre pour développer l'instruction.

ANALYSE SYNTAXIQUE DESCENDANTE

ANALYSE PRÉDICTIVE NON RÉCURSIVE

- Une pile contient une séquence de symboles grammaticaux avec le symbole \$ marquant le fond de la pile.
- Utiliser une table d'analyse dans laquelle l'analyseur va chercher la production à appliquer pour développer un non-terminal.
- Utiliser un tampon d'entrée contenant la chaîne à analyser. Cette dernière se termine par un marqueur de fin de chaîne (\$).
- Un flot de sortie.



ANALYSE SYNTAXIQUE DESCENDANTE

ANALYSE PRÉDICTIVE NON RÉCURSIVE

- Tableau d'analyse à deux dimensions $M[A, a]$, A est non-terminal, a est terminal ou \$.
- Initialement la pile contient l'axiome de la grammaire au dessus du symbole \$.
- X est le symbole au sommet de la pile et a le symbole d'entrée courant déterminent l'action de l'analyseur. Trois possibilités :
 - Si $X = a = \$$, Arrêt de l'analyseur et fin réussi de l'analyse.
 - Si $X = a \neq \$$, l'analyseur enlève X de la pile et avance son pointeur vers le symbole suivant.
 - Si X est un non-terminal, le programme consulte l'entrée $M[X, a]$ de la table d'analyse M qui est soit une X -production soit une erreur.
 - Si X -production, remplacer X au sommet de la pile par la X -production (*).
 - Si erreur, appeler la procédure de récupération sur erreur.

(*) $M[X, a] = \{X \rightarrow UVW\}$, remplacer X au sommet de la pile par WVU (avec U au sommet)

ANALYSE SYNTAXIQUE DESCENDANTE

ALGORITHME D'ANALYSE PRÉDICTIVE NON RÉCURSIVE

DONNÉES : Une chaîne w et une table d'analyse M pour une grammaire G

RÉSULTAT : Si w est dans $\mathcal{L}(G)$, une dérivation gauche pour w , erreur sinon.

Positionner le pointeur source ps sur le premier symbole de $w\$$;

```
WHILE  $X \neq \$$  DO
    soit  $X$  le symbole en sommet de la pile et  $a$  le symbole pointé par  $ps$ :
    IF  $X$  est un terminal THEN
        IF  $X = a$  THEN
            enlever  $X$  de la pile et avancer  $ps$ ;
        ELSE
            error ();
        END IF
    ELSE /*  $X$  est un non-terminal */
        IF  $M[X, a] = X \rightarrow Y_1 Y_2 \dots Y_k$  THEN
            placer  $Y_k, Y_{k-1}, \dots, Y_1$  sur la pile avec  $Y_1$  au sommet;
            émettre en sortie la production  $X \rightarrow Y_1 Y_2 \dots Y_k$ 
        ELSE
            error ();
        END IF
    END IF
END WHILE
```

ANALYSE SYNTAXIQUE DESCENDANTE

$$\begin{aligned} S &\rightarrow aSbT \mid cT \mid d \\ T &\rightarrow aT \mid bS \mid c \end{aligned}$$

accbabd \$?

Pile	Entrée	Sortie
\$S	accbabd\$	
\$TbSa	accbabd\$	$S \rightarrow aSbT$
\$TbS	ccbabd\$	
\$TbTc	ccbabd\$	$S \rightarrow cT$
\$TbT	cbabd\$	
\$Tbc	cbabd\$	$T \rightarrow c$
\$Tb	babd\$	
\$T	abd\$	
\$Ta	abd\$	$T \rightarrow aT$
\$T	bd\$	
\$Sb	bd\$	$T \rightarrow d$
\$S	d\$	
\$d	d\$	
\$	\$	

ANALYSE SYNTAXIQUE DESCENDANTE

CONSTRUCTION DE LA TABLE D'ANALYSE

- Deux fonctions Premier et Suivant , associées à une grammaire G , permettent de remplir les entrées de la table d'analyse prédictive pour G .
- Premier : Si α est une chaîne de symboles grammaticaux, $\text{Premier}(\alpha)$ désigne l'ensemble* des terminaux qui commencent par les chaînes qui se dérivent de α . Si $\alpha \Rightarrow \varepsilon$, alors ε est aussi dans $\text{Premier}(\alpha)$.
- Suivant : Pour chaque non terminal A , $\text{Suivant}(A)$ est l'ensemble de terminaux a qui peuvent apparaître immédiatement à droite de A . C'est-à-dire qu'il existe une dérivation de la forme $S \Rightarrow \alpha A a \beta$ où α et β sont des chaînes de symboles grammaticaux.

ANALYSE SYNTAXIQUE DESCENDANTE

CALCUL DE PREMIER (X)

➤ 1^{er} cas:

Si $X \rightarrow a Y_1 \mid b Y_2 \mid c Y_3 \mid \varepsilon$

Alors $P(X) = \{ a, b, c, \varepsilon \}$ avec $Y_1, Y_2, Y_3 \in \{V_t \cup V_n\}^*$

➤ 2^{eme} cas:

Si $X \rightarrow AB$ et $A, B \in V_n$

Alors Ajouter $P(A)$ sauf ε dans $P(X)$

ANALYSE SYNTAXIQUE DESCENDANTE

EXEMPLE DE CALCUL DE PREMIER (X)

$$E \rightarrow T E'$$

$$E' \rightarrow + T E' \mid \varepsilon$$

$$T \rightarrow F T'$$

$$T' \rightarrow * F T' \mid \varepsilon$$

$$F \rightarrow (E) \mid \mathbf{id}$$

- $\text{Premier}(E) = \text{Premier}(F) = \text{Premier}(T) = \{ (, \mathbf{id} \}$
- $\text{Premier}(E') = \{ +, \varepsilon \}$
- $\text{Premier}(T') = \{ *, \varepsilon \}$

ANALYSE SYNTAXIQUE DESCENDANTE

CALCUL DE SUIVANT (X)

- Mettre $\$$ dans $\text{Suivant}(S)$ où S est l'axiome et $\$$ est le marqueur de fin du texte source.
- S'il y a une production $A \rightarrow \alpha B \beta$, ajouter le contenu de $\text{Premier}(\beta)$ à $\text{Suivant}(B)$ exception faite de ε .
- Si $\beta \rightarrow \varepsilon$ ou $X \rightarrow \alpha B$ Alors ajouter $\text{Suivant}(X)$ dans $\text{Suivant}(B)$

ANALYSE SYNTAXIQUE DESCENDANTE

EXEMPLE DE CALCUL DE SUIVANT (X)

$$\begin{aligned} E &\rightarrow T E' \\ E' &\rightarrow + T E' \mid \varepsilon \\ T &\rightarrow F T' \\ T' &\rightarrow * F T' \mid \varepsilon \\ F &\rightarrow (E) \mid \text{id} \end{aligned}$$

$\text{Suivant}(E) = \{), \$\}$

- $\text{Suivant}(T) = \{\text{Premier}(E') \setminus \{\varepsilon\} \text{ et } E' \rightarrow \varepsilon \text{ donc on ajoute suivant}(E)\} = \{+,), \$\}$
- $\text{Suivant}(F) = \{\text{Premier}(T') \setminus \{\varepsilon\} \text{ et } T' \rightarrow \varepsilon \text{ donc on ajoute suivant}(T)\} = \{+, *,), \$\}$
- $\text{Suivant}(T) = \text{Suivant}(T) = \{+,), \$\}$
- $\text{Suivant}(E') = \text{Suivant}(E) = \{), \$\}$

ANALYSE SYNTAXIQUE DESCENDANTE

CONSTRUCTION DES TABLE D'ANALYSEURS PRÉDICTIFS

DONNÉES : Une grammaire G

RÉSULTAT : Une table d'analyse M pour G .

- 1) Pour chaque production $A \rightarrow \alpha$ de la grammaire, procéder aux étapes 2 et 3 suivantes :
- 2) Pour chaque terminal a dans $\text{Premier}(\alpha)$, ajouter $A \rightarrow \alpha$ à $M[A, a]$.
- 3) Si $\varepsilon \in \text{Premier}(\alpha)$, ajouter $A \rightarrow \alpha$ à $M[A, b] \forall b \in \text{Suivant}(A)$ et b est terminal.
Si $\varepsilon \in \text{Premier}(\alpha)$ et $\$ \in \text{Suivant}(A)$, ajouter $A \rightarrow \alpha$ dans $M[A, \$]$.
- 4) Faire de chaque entrée non définie de M une erreur.

ANALYSE SYNTAXIQUE DESCENDANTE

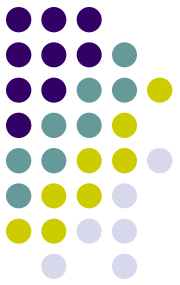
EXEMPLE DE CONSTRUCTION DE TABLE PRÉDICTIVE

$E \rightarrow T E'$
 $E' \rightarrow + T E' \mid \varepsilon$
 $T \rightarrow F T'$
 $T' \rightarrow * F T' \mid \varepsilon$
 $F \rightarrow (E) \mid id$

- $Premier(E) = Premier(F) = Premier(T) = \{ (, id \}$
- $Premier(E') = \{ +, \varepsilon \}$
- $Premier(T') = \{ *, \varepsilon \}$
- $Suivant(E) = Suivant(E') = \{ \}, \$ \}$
- $Suivant(T) = Suivant(T') = \{ +, \varepsilon, \$ \}$
- $Suivant(F) = \{ +, *, \varepsilon, \$ \}$

$E \rightarrow T E'$
 $E' \rightarrow + T E' \mid \varepsilon$
 $T \rightarrow F T'$
 $T' \rightarrow * F T' \mid \varepsilon$
 $F \rightarrow (E) \mid id$

NON-TERMINAL	SYMBOLE D'ENTRÉE					
	id	+	*	()	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow id$			$F \rightarrow (E)$		



ANALYSE SYNTAXIQUE DESCENDANTE

id+id*id\$?

Pile	Entrée	Sortie
\$E	id+id*id\$	
\$E'T	id+id*id\$	$E \rightarrow TE'$
\$E'T'F	id+id*id\$	$T \rightarrow FT'$
\$E'T'id	id+id*id\$	$F \rightarrow id$
\$E'T'	+id*id\$	
\$E'	+id*id\$	$T' \rightarrow \epsilon$
\$E'T+	+id*id\$	$E' \rightarrow +TE'$

Pile	Entrée	Sortie
\$E'T	id*id\$	
\$E'T'F	id*id\$	$T \rightarrow FT'$
\$E'T'id	id*id\$	$F \rightarrow id$
\$E'T'	*id\$	
\$E'T'F*	*id\$	$T' \rightarrow *FT'$
\$E'T'F	id\$	
\$E'T'id	id\$	$F \rightarrow id$

Pile	Entrée	Sortie
\$E'T'	\$	
\$E'	\$	$T' \rightarrow \epsilon$
\$	\$	$E' \rightarrow \epsilon$

GRAMMAIRE LL(1)

DÉFINITION

- Pour certaines grammaires, la table M peut avoir des entrées définies de façon multiple.
- La grammaire dont la table d'analyse n'a aucune entrée définie de façon multiple est dite LL(1).
- **L** : « Left to right scanning » (parcours de l'entrée de gauche vers l'entrée de droite), **L** : « Leftmost derivation » (dérivation à gauche), **1** : On utilise un seul symbole d'entrée de prévision à chaque étape nécessitant la prise d'une décision d'action d'analyse.

Une grammaire est LL(1) \Leftrightarrow

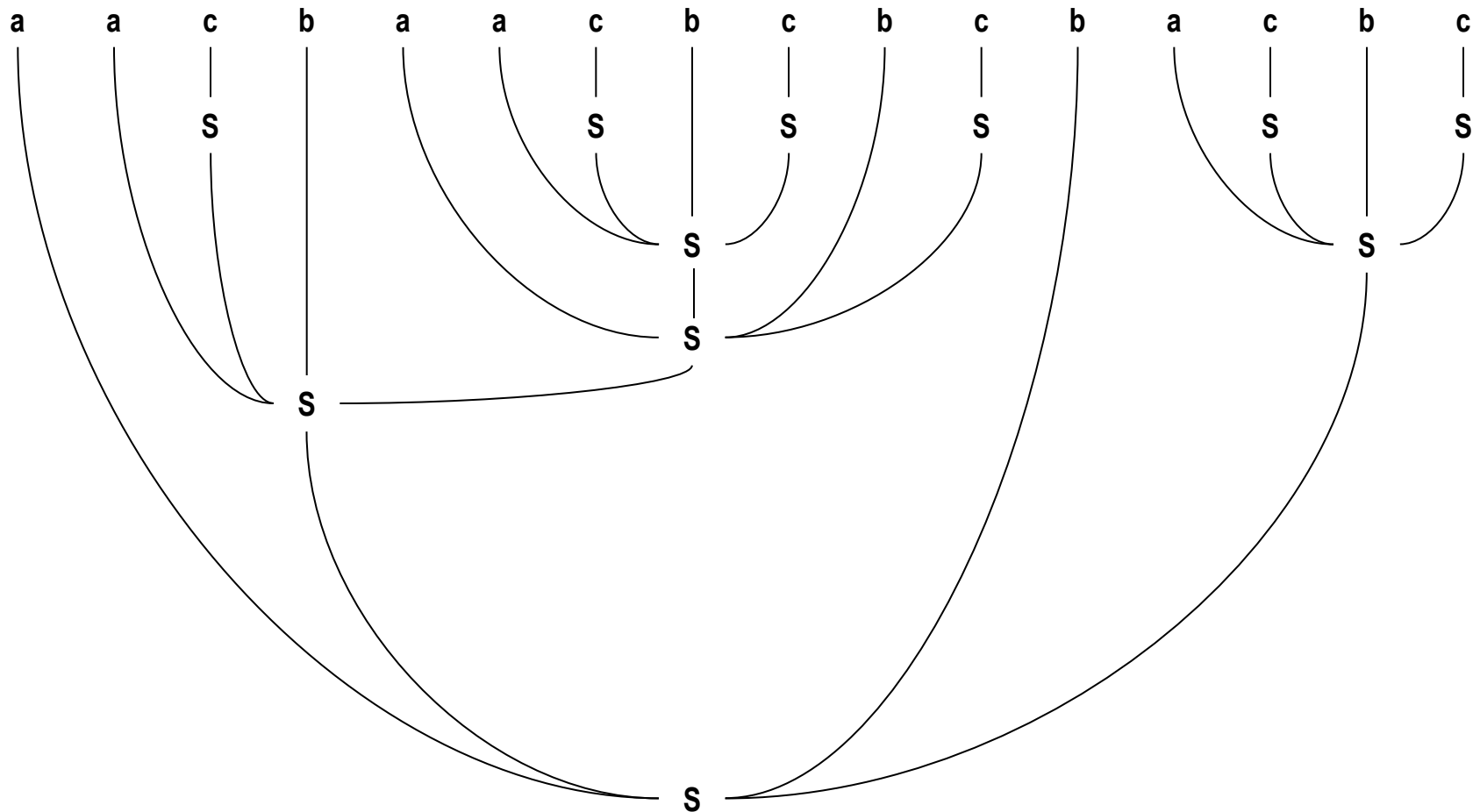
- factorisée
- Non réursive à gauche
- Non ambiguë

ANALYSE SYNTAXIQUE ASCENDANTE

- Aussi connue comme « Analyse par décalage réduction ».
- Construire un arbre d'analyse pour une chaîne source en commençant par les feuilles.
- Peut être considérée comme la réduction d'une chaîne w vers l'axiome de la grammaire.
- A chaque étape de réduction, une sous-chaîne particulière correspondant à la partie droite d'une production est remplacée par le symbole de la partie gauche

ANALYSE SYNTAXIQUE ASCENDANTE

EXEMPLE

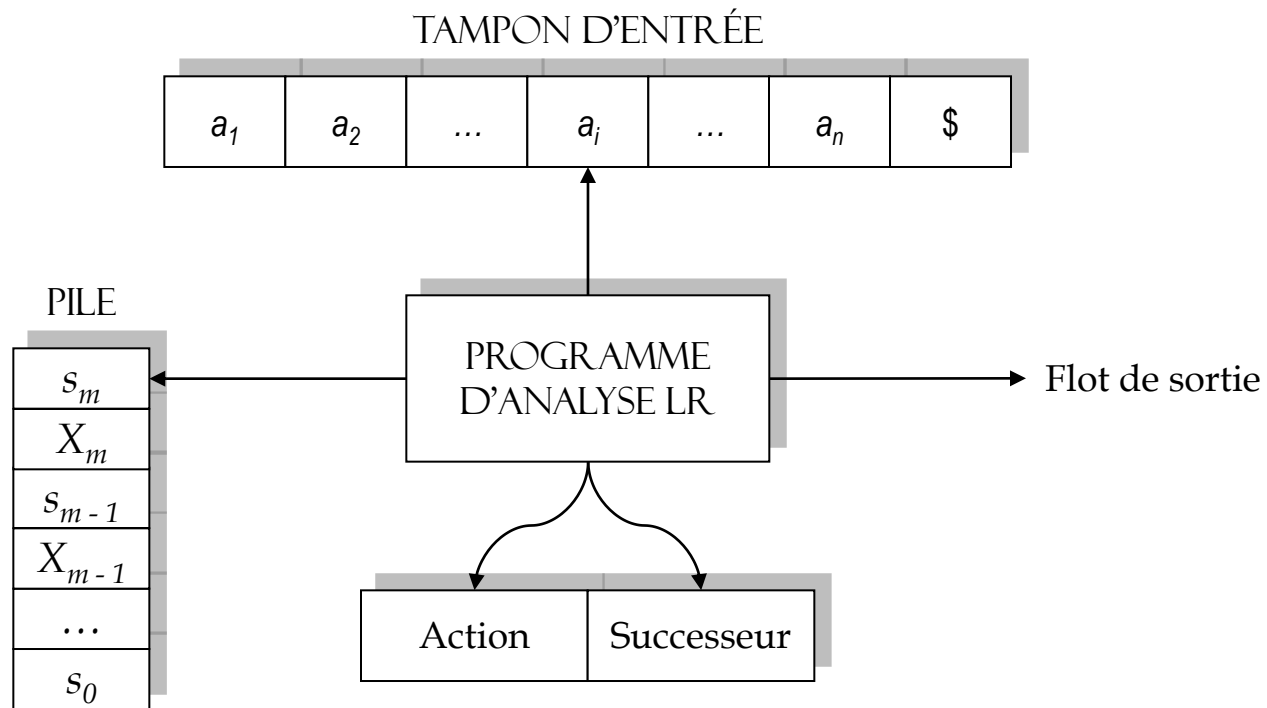


ANALYSEURS LR

- Analyseurs syntaxique ascendants.
- **L** : « Left to right scanning » (parcours de l'entrée de gauche vers l'entrée de droite).
- **R** : « Constructing a Rightmost derivation in reverse » (en construisant une dérivation droite inverse).
- **k** : k symboles en prévision utilisés pour prendre les décisions d'analyse (par défaut $k = 1$).
- Analyse toute grammaire non contextuelle.
- Implémentation efficace.
- Détection des erreurs de syntaxe aussitôt que possible.

ANALYSEURS LR

MODÈLE D'ANALYSEUR LR



ANALYSEURS LR

MODÈLE D'ANALYSEUR LR

- Les tables d'analyse sont subdivisées en deux parties (*Action* et *successeur*).
- Le programme moteur est le même pour tous les analyseurs LR.
- Les tables d'analyses changent d'un analyseur à l'autre.
- Le programme d'analyse lit les unités lexicales les unes après les autres dans un tampon.
- Il utilise une pile pour y ranger les chaînes $s_0 X_1 s_1 X_2 s_2 \dots X_m s_m$ où s_m est au sommet. X_i est un symbole de la grammaire et s_i est symbole appelé état.
- Chaque état résume l'information de la pile en dessous de lui.
- La combinaison : état en sommet de la pile et le symbole d'entrée courant est utilisée pour indiquer les tables et déterminer l'action d'analyse *décaler* ou *réduire* à effectuer.

ANALYSEURS LR

ALGORITHME D'ANALYSE LR

- Le programme d'analyse lit l'état s_m en sommet de la pile et a_i le symbole terminal d'entrée courant.
- Il consulte *Action* [s_m , a_i], l'entrée de la table des actions pour l'état s_m et le terminal a_i . Cela peut avoir l'une des quatre valeurs :
 1. *décaler* s , où s est un état.
 2. *réduire* par une production $A \rightarrow \beta$ de la grammaire.
 3. *accepter*
 4. *erreur*
- La fonction *Successeur* prend comme arguments un état et un symbole non terminal et retourne un état.

ANALYSEURS LR

ALGORITHME D'ANALYSE LR

- Configuration d'un analyseur LR: un couple dont le premier composant est le contenu de la pile et dont le second composant est la chaîne d'entrée restant à analyser.

$$(s_0 X_1 s_1 X_2 s_2 \dots X_m s_m, a_i a_{i+1} \dots a_n \$)$$

Cette configuration représente la proto phrase droite $X_1 X_2 \dots X_m a_i a_{i+1} \dots a_n$

- L'action suivante est déterminée par la lecture de a_i le symbole d'entrée courant, s_m , l'état en sommet de pile et *Action* $[s_m, a_i]$ de la table des actions d'analyse.
- Il en résulte quatre configurations après chaque type d'actions.

ANALYSEURS LR

ALGORITHME D'ANALYSE LR

Action décaler

Si $Action[s_m, a_i] = \text{décaler } s$, l'analyseur exécute une action décaler, atteignant la configuration :

$$(s_0 X_1 s_1 X_2 s_2 \dots X_m s_m \textcolor{red}{a_i} \textcolor{red}{s}, a_{i+1} \dots a_n \$)$$

L'analyseur a à la fois empilé le symbole d'entrée courant a_i et le prochain état s qui est donné par $Action[s_m, a_i]$; a_{i+1} devient le symbole d'entrée courant.

ANALYSEURS LR

ALGORITHME D'ANALYSE LR

Action réduire

Si $Action[s_m, a_i] = \text{réduire}$ par $A \rightarrow \beta$, alors l'analyseur exécute une action *réduire* atteignant la configuration suivante :

$$(s_0 X_1 s_1 X_2 s_2 \dots X_{m-r} s_{m-r} A s, a_i a_{i+1} \dots a_n \$)$$

$s = Successeur[s_{m-r}, A]$ et r est la longueur de β . L'analyseur commence par dépiler $2r$ symboles (r symboles d'états et r symboles grammaticaux), exposant l'état s_{m-r} au sommet.

L'analyseur empile A la partie gauche de la production et s l'entrée pour $Successeur[s_{m-r}, A]$.

Le symbole courant en entrée n'est pas changé par une action réduire.

ANALYSEURS LR

ALGORITHME D'ANALYSE LR

Action Accepter

Si *Action* $[s_m, a_i] = \text{accepter}$, l'analyse est terminée.

Action Erreur

Si *Action* $[s_m, a_i] = \text{erreur}$, l'analyseur a découvert une erreur et appelle une routine de récupération sur erreur.